Efficient Algorithms for Demand-Aware Networks and a Connection to Virtual Network Embedding

Aleksander Figiel[1] Janne H. Korhonen[1]
Neil Olver[2] Stefan Schmid[1,3]

[1]TU Berlin, Germany

[2]London School of Economics and Political Science, UK

[3]Fraunhofer SIT

Datacenter networks

Bandwidth requirements in datacenters are growing explosively!

Datacenter networks

Bandwidth requirements in datacenters are growing explosively!

Classic datacenter topologies:

- oblivious to the traffic / demand they serve
- typically optimized for some worst case metric, e.g.: full bisection bandwidth

Demand-aware networks (DANs):

- optimized for the traffic / demand they serve

Formal problem

Introduced by Avin et al. (DISC 2017)

Bounded Degree Demand-Aware Network Design (BND)

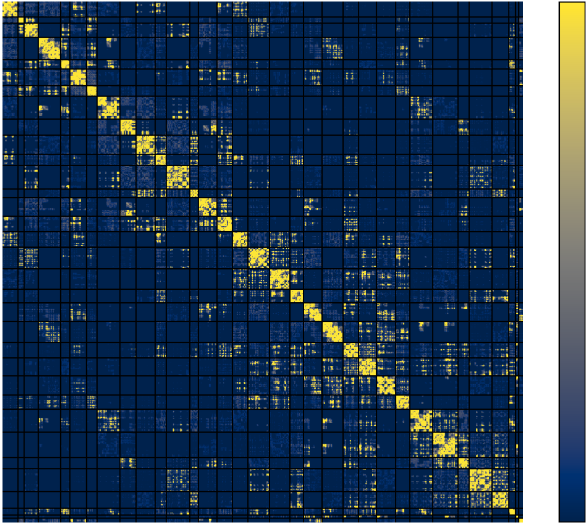
Input: A degree bound $\Delta \in \mathbb{N}$,
a node set $V = \{1, 2, \ldots, n\}$, and
a probability distribution $p \colon \binom{V}{2} \to [0, 1]$ (the demand)

Task: Find an undirected graph $G = (V, E)$ (the host) with maximum degree at most Δ that minimizes the *expected path length* (EPL):

$$\sum_{\{u,v\} \in \binom{V}{2}} p(\{u, v\}) d_G(u, v)$$

- $d_G(u, v)$ - shortest path distance

Traffic matrix



Formal problem

For $\Delta = 2$ it is essentially Minimum Linear Arrangement [Avin et al., DISC 2017]

[1] decision version, positive integer weights instead of probabilities

Formal problem

For $\Delta = 2$ it is essentially Minimum Linear Arrangement [Avin et al., DISC 2017]

We show NP-completeness for BND[1]:

- $\Delta = 2$: reduction from Undirected Circular Arrangement
- $\Delta \geq 3$: reduction from Vertex Cover on 3-regular graphs

[1] decision version, positive integer weights instead of probabilities

Formal problem

For $\Delta = 2$ it is essentially Minimum Linear Arrangement [Avin et al., DISC 2017]

We show NP-completeness for BND[1]:

- $\Delta = 2$: reduction from Undirected Circular Arrangement
- $\Delta \geq 3$: reduction from Vertex Cover on 3-regular graphs

$\implies$: main focus on approximation algorithms and heuristics

[1] decision version, positive integer weights instead of probabilities

Known results

The demand distribution can be thought of as a demand graph D

- only edges for which $p(\{u, v\}) > 0$

Entropy based lower bound:

- EPL $\geq \frac{1}{2} H_{d+1}(D)$ [Avin et al., DISC 2017]

Approximation for sparse demands:

- $\mathcal{O}(\log \Delta)$-approximation when $\Delta = 12\Delta_{\text{avg}}(D)$ [Avin et al., DISC 2017]
- $\mathcal{O}(\log \Delta)$-approximation when $\Delta = 2\rho(D)$ [Peres and Avin, INFOCOM 2023]
- $\mathcal{O}(1)$-approximation when $\Delta = 4\Delta_{\text{avg}}(D) + 1$ [**this work**]

---

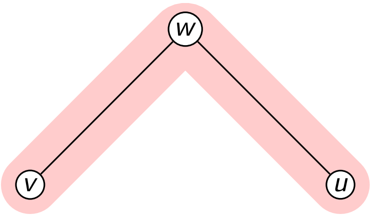
[2] $H_{d+1}(D)$ - base-$d+1$ conditional entropy
[3] $\Delta_{\text{avg}}(D)$ - average degree
[4] $\rho(D)$ - arboricity

Static vs demand-aware

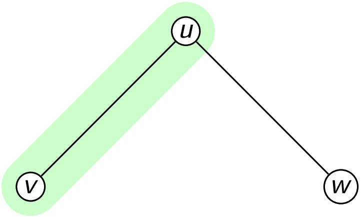
If network topology is fixed, is node relocation (permutation) sufficent?

Static vs demand-aware

If network topology is fixed, is node relocation (permutation) sufficent?
Example: assume $p(\{u, v\})$ is very high.



2 hops & high stretch



1 hop & low stretch

No universal host graph

Theorem (Simplified)

For any host graph $G = (V, E)$ of size n and maximum degree Δ, there exists some demand graph D for which the optimal embedding into G is only a $\Omega(\log \log n)$ approximation of an optimal host graph.

$\implies$ VNEP is not a good approximation for BND

No universal host graph

Theorem (Simplified)

For any host graph $G = (V, E)$ of size n and maximum degree Δ, there exists some demand graph D for which the optimal embedding into G is only a $\Omega(\log \log n)$ approximation of an optimal host graph.

$\implies$ VNEP is not a good approximation for BND

The demand graph also has maximum degree Δ.

$\implies$ optimal host graph is the demand graph!

Balancing sparse demands

Balancing algorithm:

Let $\Delta_{\text{avg}} = \Delta_{\text{avg}}(D)$

Split nodes into two groups:

- high degree: $\deg_D(v) > 2\Delta_{\text{avg}}$
- low degree: $\deg_D(v) \leq 2\Delta_{\text{avg}}$

At most $n/2$ high degree nodes.

Balancing sparse demands

Balancing algorithm:
Let $\Delta_{\text{avg}} = \Delta_{\text{avg}}(D)$
Split nodes into two groups:

- high degree: $\deg_D(v) > 2\Delta_{\text{avg}}$
- low degree: $\deg_D(v) \leq 2\Delta_{\text{avg}}$

At most $n/2$ high degree nodes.

For each node v:

- First $2\Delta_{\text{avg}}$ edges with highest probability are *high demand* edges for v
- Remaining edges are *low demand* edges for v
- Build $2\Delta_{\text{avg}}$-ary Huffman tree T_v with terminal set L_v and v as root

Balancing sparse demands

Balancing algorithm:

Let $\Delta_{\text{avg}} = \Delta_{\text{avg}}(D)$

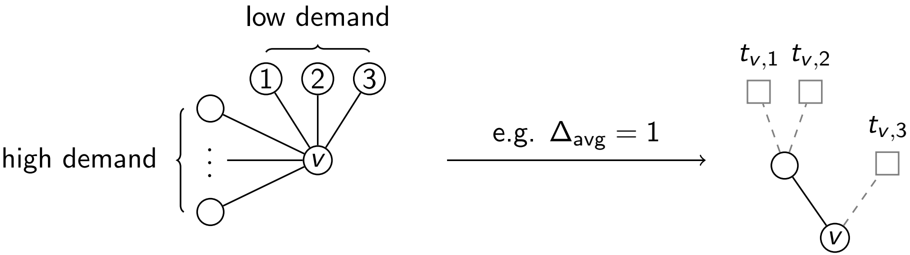
Split nodes into two groups:

- high degree: $\deg_D(v) > 2\Delta_{\text{avg}}$
- low degree: $\deg_D(v) \leq 2\Delta_{\text{avg}}$

At most $n/2$ high degree nodes.

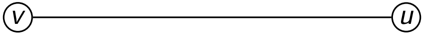
For each node v:

- First $2\Delta_{\text{avg}}$ edges with highest probability are *high demand* edges for v
- Remaining edges are *low demand* edges for v
- Build $2\Delta_{\text{avg}}$-ary Huffman tree T_v with terminal set L_v and v as root



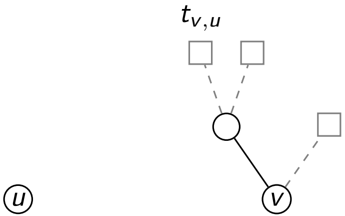
Connecting the pieces

For every edge $\{u, v\}$ in the demand graph add the following edge in the host graph:
Case 1: $\{u, v\}$ is high demand for both u, v



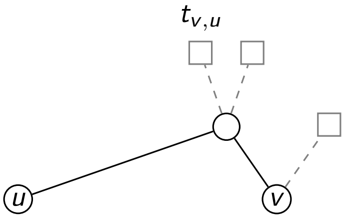
Connecting the pieces

For every edge $\{u, v\}$ in the demand graph add the following edge in the host graph:
Case 2: $\{u, v\}$ is high demand for only one, w.l.o.g. u



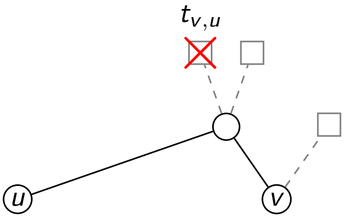
Connecting the pieces

For every edge $\{u, v\}$ in the demand graph add the following edge in the host graph:
Case 2: $\{u, v\}$ is high demand for only one, w.l.o.g. u



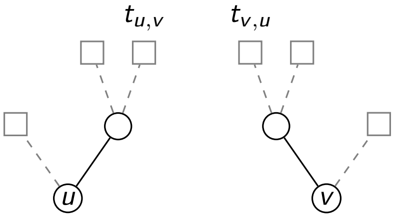
Connecting the pieces

For every edge $\{u, v\}$ in the demand graph add the following edge in the host graph:
Case 2: $\{u, v\}$ is high demand for only one, w.l.o.g. u



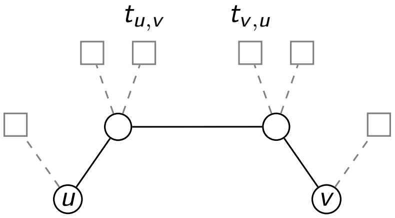
Connecting the pieces

For every edge $\{u, v\}$ in the demand graph add the following edge in the host graph:
Case 3: $\{u, v\}$ is low demand for u, v



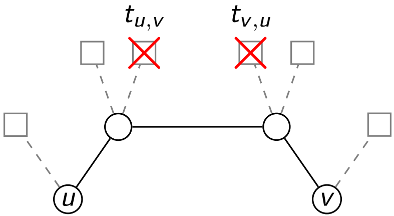
Connecting the pieces

For every edge $\{u, v\}$ in the demand graph add the following edge in the host graph:
Case 3: $\{u, v\}$ is low demand for u, v



Connecting the pieces

For every edge $\{u, v\}$ in the demand graph add the following edge in the host graph:
Case 3: $\{u, v\}$ is low demand for u, v



Analysis

What about inner nodes of Huffman trees?
Map injectively to low degree nodes!
By Markov's inequality at most $n/2$ high degree nodes.

Analysis

What about inner nodes of Huffman trees?
Map injectively to low degree nodes!
By Markov's inequality at most $n/2$ high degree nodes.
Node degrees
Low degree nodes:

- At most $2\Delta_{\text{avg}}$ direct edges
- At most $2\Delta_{\text{avg}} + 1$ Huffman tree edges (internal only)

Analysis

What about inner nodes of Huffman trees?

Map injectively to low degree nodes!

By Markov's inequality at most $n/2$ high degree nodes.

Node degrees

Low degree nodes:

- At most $2\Delta_{\text{avg}}$ direct edges
- At most $2\Delta_{\text{avg}} + 1$ Huffman tree edges (internal only)

High degree nodes:

- At most $2\Delta_{\text{avg}}$ direct edges
- At most $2\Delta_{\text{avg}}$ Huffman tree edges (root only)

Analysis

What about inner nodes of Huffman trees?

Map injectively to low degree nodes!

By Markov's inequality at most $n/2$ high degree nodes.

Node degrees

Low degree nodes:

- At most $2\Delta_{\text{avg}}$ direct edges
- At most $2\Delta_{\text{avg}} + 1$ Huffman tree edges (internal only)

$\implies \Delta \leq 4\Delta_{\text{avg}} + 1$

High degree nodes:

- At most $2\Delta_{\text{avg}}$ direct edges
- At most $2\Delta_{\text{avg}}$ Huffman tree edges (root only)

Analysis

What about inner nodes of Huffman trees?
Map injectively to low degree nodes!
By Markov's inequality at most $n/2$ high degree nodes.

Node degrees

Low degree nodes:

- At most $2\Delta_{\text{avg}}$ direct edges
- At most $2\Delta_{\text{avg}} + 1$ Huffman tree edges (internal only)

$\implies \Delta \leq 4\Delta_{\text{avg}} + 1$

High degree nodes:

- At most $2\Delta_{\text{avg}}$ direct edges
- At most $2\Delta_{\text{avg}}$ Huffman tree edges (root only)

EPL

Using entryopy based lower-bound [Avin et al., DISC 2017] and properties of Huffman trees: $\implies$ constant factor approximation

Results on real data

Trace	Split&Join		Helper		**Balancing**	
	Δ	EPL	Δ	EPL	Δ	EPL
fb-cluster-rack-A	507	1.33	399	3.26	290	1.31
fb-cluster-rack-B	1,305	1.06	855	4.25	754	1.05
fb-cluster-rack-C	1,460	1.14	870	3.51	683	1.08
hpc-cesar-mocfe	21	1.33	18	2.87	21	1.0
hpc-cesar-nekbone	39	1.07	36	1	36	1
hpc-boxlib-cns	157	1.11	102	1.04	293	1.01
hpc-boxlib-multigrid	31	1.12	26	1	26	1
p-fabric-trace-0-1	143	1.01	144	1	144	1
p-fabric-trace-0-5	143	1.01	143	1	143	1
p-fabric-trace-0-8	143	1.01	143	1	143	1

[5] Split&Join - based on graph arboricity [Peres and Avin, INFOCOM 2023]
[6] Helper - based on average degree [Avin et al., DISC 2017]

What about arbitrary Δ?

BND with Steiner Nodes

BOUNDED NETWORK DESIGN WITH STEINER NODES

Input: A degree bound $\Delta \in \mathbb{N}$,
a node set $V = \{1, 2, \ldots, n\}$, and
a probability distribution $p\colon \binom{V}{2} \to [0, 1]$ (the demand)

Task: Find an undirected graph $G = (V \cup U)$ (the host) with maximum degree at most Δ that minimizes the *expected path length* (EPL):

$$\sum_{\{u,v\} \in \binom{V}{2}} p(\{u, v\}) d_G(u, v)$$

BND with Steiner Nodes

Bounded Network Design with Steiner Nodes

Input: A degree bound $\Delta \in \mathbb{N}$,
a node set $V = \{1, 2, \ldots, n\}$, and
a probability distribution $p\colon \binom{V}{2} \to [0, 1]$ (the demand)

Task: Find an undirected graph $G = (V \cup U)$ (the host) with maximum degree at most Δ that minimizes the *expected path length* (EPL):

$$\sum_{\{u,v\} \in \binom{V}{2}} p(\{u, v\}) d_G(u, v)$$

Sadly, also NP-complete for all $\Delta \geq 2$.

Constant factor approximation for $\Delta \geq 3$

Similiar idea to the approximation on sparse graphs.

Constant factor approximation for $\Delta \geq 3$

Similiar idea to the approximation on sparse graphs.

Step 1: Build trees

For every node v build a $(\Delta - 1)$-ary Huffman tree on the terminal set $N(v)$.

Constant factor approximation for $\Delta \geq 3$

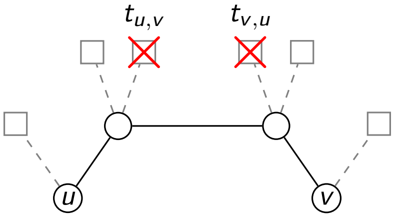
Similiar idea to the approximation on sparse graphs.

Step 1: Build trees

For every node v build a $(\Delta - 1)$-ary Huffman tree on the terminal set $N(v)$.

Step 2: Connect trees

For every edge $\{u, v\}$ in the demand:



Constant factor approximation for $\Delta \geq 3$

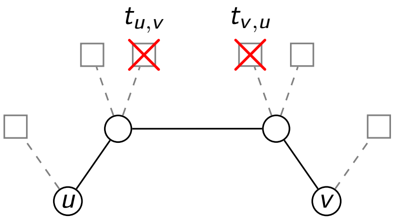
Similiar idea to the approximation on sparse graphs.

Step 1: Build trees

For every node v build a $(\Delta - 1)$-ary Huffman tree on the terminal set $N(v)$.

Step 2: Connect trees

For every edge $\{u, v\}$ in the demand:



Step 3: Finalize

Internal nodes of Huffman trees become Steiner nodes

Constant factor approximation for $\Delta \geq 3$

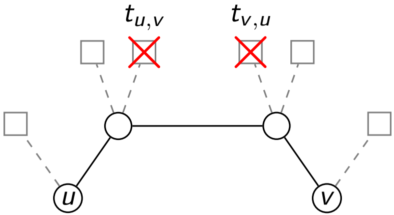
Similiar idea to the approximation on sparse graphs.

Step 1: Build trees

For every node v build a $(\Delta - 1)$-ary Huffman tree on the terminal set $N(v)$.

Step 2: Connect trees

For every edge $\{u, v\}$ in the demand:



Step 3: Finalize

Internal nodes of Huffman trees become Steiner nodes

Analysis: Very similar to previous result!

Conclusion

Classification:

- NP-completness for Bounded Network Design (also with Steiner nodes) for all $\Delta \geq 2$

VNEP:

- no universal host graph theorem

Approximation:

- constant factor approximation for BND on sparse demands
- constant factor approximation for BND with Steiner nodes ($\Delta \geq 3$)

Conclusion

Classification:

- NP-completness for Bounded Network Design (also with Steiner nodes) for all $\Delta \geq 2$

VNEP:

- no universal host graph theorem

Approximation:

- constant factor approximation for BND on sparse demands
- constant factor approximation for BND with Steiner nodes ($\Delta \geq 3$)

Thank you!